We Claim:

1. A method for monitoring user actions on a computer system, comprising:

(a) determining, with a first application programming interface (API), whether a first screen object has been acted upon by a user, the first API being coordinate-independent and application message independent with respect to the first screen object; and

(b) in response to (a), capturing a user event associated with the first screen object.

2. The method of claim 1, further comprising:

(c) processing the captured user event.

3. The method of claim 1, wherein the first API comprises an Active Accessessibility® API.

4. The method of claim 1, further comprising:

(d) determining, with a second API, whether a second screen object has been acted upon by the user.

5. The method of claim 1, further comprising:

(d) determining, with a second API, whether the first screen object has been acted upon by the user.

6. The method of claim 2, wherein (c) comprises:

(i) representing the captured user event as an event entry in a file.

7. The method of claim 6, wherein (c) further comprises:

(ii) storing the file.

8. The method of claim 7, wherein (c) further comprises:

(iii) retrieving the file.

9. The method of claim 8, wherein (c) further comprises:

(iv) playing back the user event from the event entry of the file.

10. The method of claim 6, wherein (c) further comprises:

(ii) editing the event entry of the file.

11. The method of claim 10, wherein (ii) comprises:

(1) modifying the event entry to represent a modified user event.

12. The method of claim 6, wherein the file comprises a text file.

13. The method of claim 7, wherein the text file complies with an Extensible Markup Language (XML) format.

14. The method of claim 2, further comprising:

(d) inputting a command, through a user interface, that is indicative of subsequent processing of the user event.

15. The method of claim 14, wherein the command is indicative of recording the user event, wherein (c) comprises:

(i) determining a speed associated with the user event;

(ii) determining whether a cursor is positioned over the first screen object; and

(iii) if the cursor is over the first object, accessing and recording parameters associated with the first screen object.

16. The method claim 15, wherein (c) further comprises:

(iv) highlighting the first screen object.

17. The method of claim 15, wherein (c) further comprises:

(iv) if a keystroke is entered, associating the keystroke with a previously recorded object.

18. The method of claim 7, wherein (ii) comprises:

(1) creating a knowledge base for archiving and exchanging at least one file, wherein each file comprises a representation of a set of user events.

19. The method of claim 18, wherein (ii) further comprises:

(2) maintaining the knowledge base in accordance with at least one subsequent user event.

20. The method of claim 1, wherein the first API is selected from the group consisting of an Access Accessibility® API, a Win32® API, and a Windows® system hooks API.

21. The method of claim 1, wherein the first screen object is associated with an application program.

22. The method of claim 21, wherein the first screen object comprises a desktop object.

23. The method if claim 1, wherein the first screen object is associated with a web page.

24. The method of claim 1, wherein the user event occurs on a first computer of the computer system and wherein the user event is captured on the first computer.

25. The method of claim 1, wherein the user event occurs on a first computer of the computer system and wherein the user event is captured on a second computer of the computer system.

26. The method of claim 25, wherein an application or web page interacts with a remote software component through a toolbar in conjunction with a terminal service client.

27. The method of claim 13, wherein the XML file is exported as a hyper text markup language (HTML) file, wherein a web browser is utilized to playback the HTML file.

28. The method of claim 14, wherein the command is selected from the group consisting of a new command, an open command, a view command, a save command, a notes command, a record command, a back command, and a next command.

29. The method of claim 14, wherein the command is indicative of playing back the user event, wherein (d) comprises:

(i) reading the event entry from a text file; and

(ii) reproducing the user event from the determining whether a cursor is positioned over the first screen object.

30. The method of claim 14, wherein the command is indicative of playing back a file, wherein (c) comprises:

(i) enumerating a desktop;

(ii) in response to (i), drilling down through a hierarchy to find a matching screen object in accordance with at least one attribute of the event entry; and

(iii) if the matching screen object is not found, stopping playback of the file; and

(iv) if the matching screen object is found, invoking a recorded action that is associated with the user event.

31. The method of claim 30, further comprising:

(v) in response to (iv), proceeding to a next user event that is recorded by the file.

32. The method of claim 12, wherein the event entry comprises a notes attribute, the notes attribute providing an annotation about the user event.

33. The method of claim 1, wherein (b) is performed by an ActiveX® component.

34. The method of claim 2, wherein (C) is performed by an ActiveX® component.

35. The method of claim 6, wherein the event entry comprises a text entry.

36. A computer-readable medium having computer-executable instructions for performing the method as recited in claim 1.

37. A computer-readable medium having computer-executable instructions for performing the method as recited in claim 2.

38. A computer-readable medium having computer-executable instructions for performing:

(a) a processing module that captures and processes a user event by utilizing an application programming interface (API), wherein the user event is associated with a screen object and wherein the API is coordinate-independent and application message independent with respect to the screen object; and

(b) a data storage module that converts the user event to an event entry in a file.

39. The computer-readable medium of clam 38, further comprising:

(c) an input user interface module that receives a command and notifies the processing module about the command, the command being indicative about subsequent capturing and processing of the user event by the processing module.

40. A computer-readable medium having stored thereon a data structure, comprising:

(a) a first data field that identifies an object name of a screen object that is associated with a user event;

(b) a second data field that identifies an object role of the screen object:

(c) a third data field that identifies an object class name of the screen object;

(d) a fourth data field that identifies a parent name, the parent name being associated with a parent of the screen object;

(e) a fifth data field that identifies a parent role, the parent role being associated with the parent of the screen object;

(f) a sixth data field that identifies a primer window, the primer window being a window class name being associated with a topmost window of the screen object;

(g) a seventh data field that identifies an action type, the action type being associated with a mouse action that is being recorded; and

(h) an eighth data field that identifies a keyboard input that is associated with the user event.

41. A computer-readable medium having stored thereon a data structure of claim 40, further comprising:

(i) a ninth data field that identifies textual information to be displayed during playback of the data structure.

42. A method for monitoring user actions on a computer system, comprising:

(a) inputting a command that is indicative of subsequent processing of the user event.

(b) in response to (a), determining, with an application programming interface (API), whether a screen object has been acted upon by a user, the API being coordinate-independent and application message independent with respect to the screen object;

(c) in response to (a), capturing a user event associated with the screen object;

(d) representing the captured user event as an event entry in a text file;

(e) subsequently retrieving the text file; and

(f) playing back the user event from the event entry of the text file, wherein the user event is reproduced on an output device.

43. A method of claim 1, further comprising:

(c) determining, with the first API, whether another screen object has been acted upon by the user, the first API being coordinate-independent and application message independent with respect to the other screen object; and

(d) in response to (c), capturing another user event associated with the other screen object.

44. The method of claim 1, further comprising:

(d) determining, with a second API, whether the first screen object has been acted upon by the user.